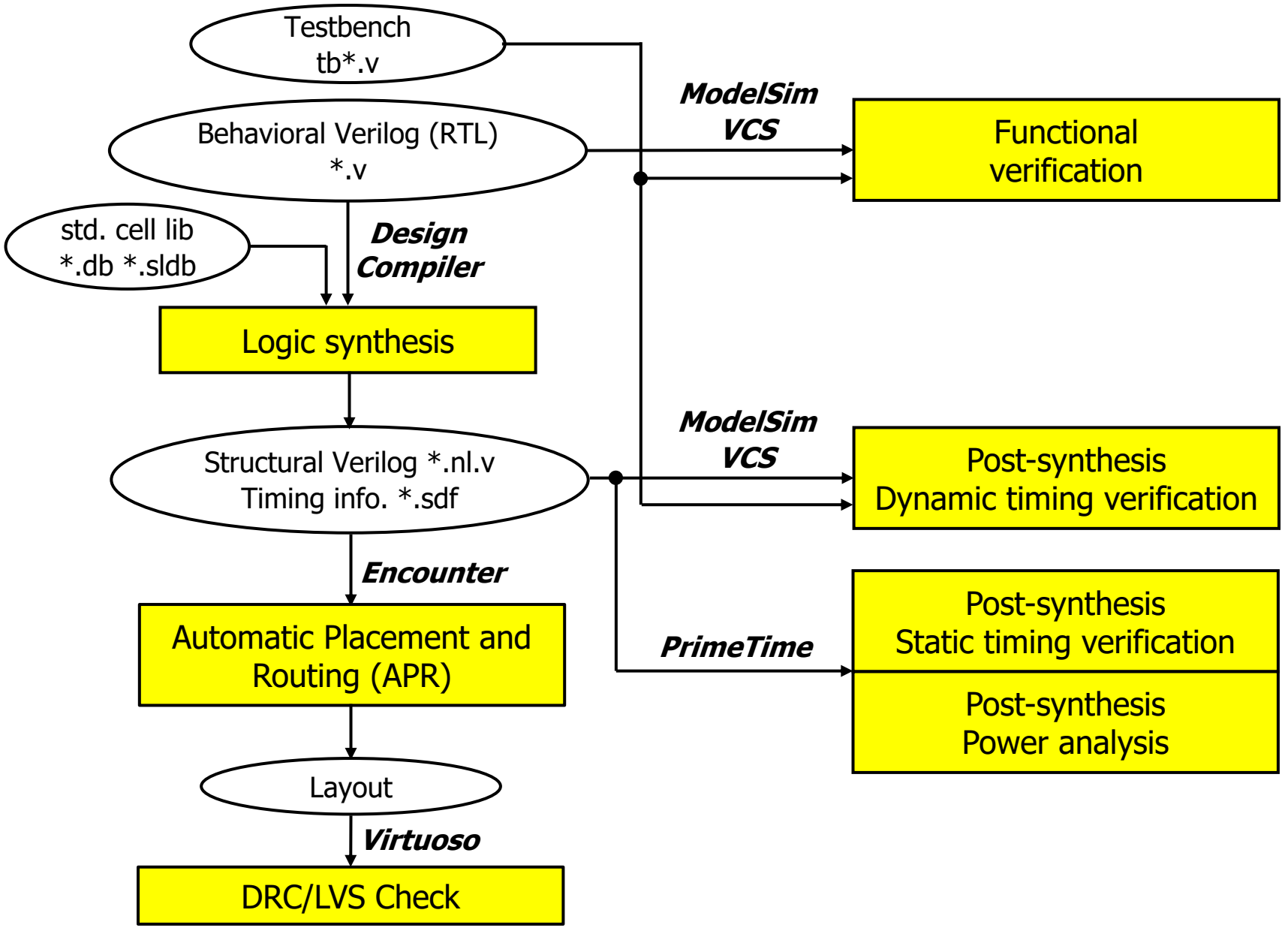
EECS E6321 Tutorial 02

Verilog, Synthesis, STA

Mingoo Seok & Previous TAs

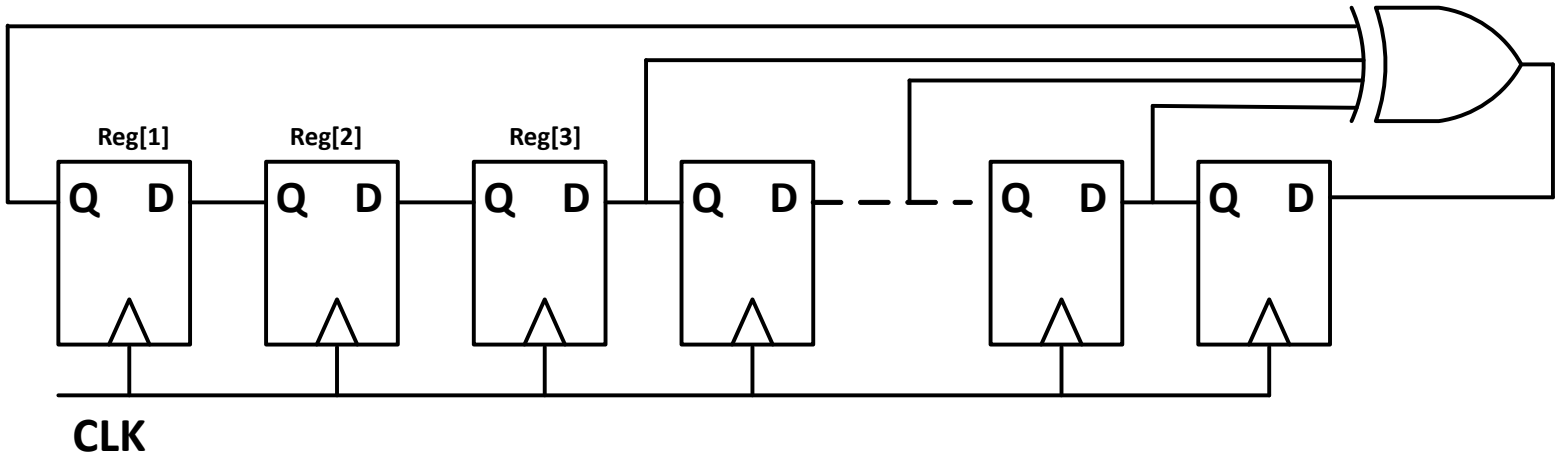
Semi-Custom Design Flow: Block-level



To start with…

- Copy files from
  - /courses/ee6350/proj_2025Spring/ref/matlab
    - lfsr1: example
  - /courses/ee6350/proj_2025Spring/ref/rtl
    - lfsr1: example
  - /courses/ee6350/proj_2025Spring/ref/qsim_rtl
    - tb_lfsr1: example
  - /courses/ee6350/proj_2025Spring/ref/dc
    - common_scripts
    - lfsr1: example
  - /courses/ee6321/share/6321-spring2021/students/qsim_dc
    - tb_lfsr1: example
  - /courses/ee6321/share/6321-spring2021/students/pt
    - lfsr1: example
- Please run your scripts at your local directories
- Maintain the same file/directory structure with the sample

Linear Feedback Shift Register (LFSR)

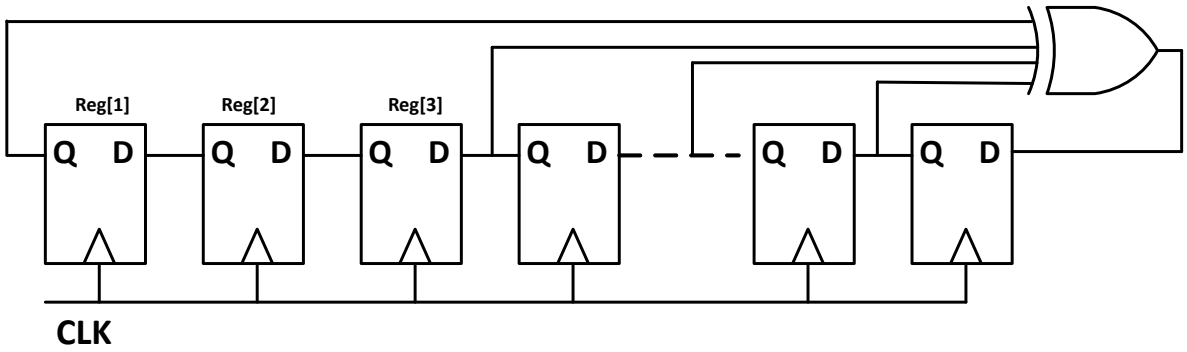


- Pseudo-random pattern generators
  - Generates periodic sequence
  - Start in a non-zero state
- FFs plus a few XOR gates

- When implementing the hardware, we need to make **Matlab or Python** model first for verifying its functionality

Matlab: lfsr1.m

```
% Behavior model for LFSR1
% Producing 256 16b outputs
% MS 7/2015

clear;

% Declare main data structures
reg = zeros(16, 1);
reg = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1];
fb = zeros(16,1);
fb = [1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1];

bw = length(fb);
n_cycle = 256;
out_file = fopen('./lfsr1_matlab.result','w');

z(1) = sum(reg.*2.^(numel(reg)-1:-1:0)); % z(1) = reg[1]*2^15 + reg[2]*2^15 + .... + reg[15]*1 + reg[16] // element-wise calculation

for i = 1:n_cycle
   fb_sum_out = 0;
   xor_out = 0;

    for j = 1:bw % "fb_sum_out" counts the number of 1s in the feedback taps
      if fb(j) == 1
         fb_sum_out = fb_sum_out + reg(j);
      end
   end

   xor_out = mod(fb_sum_out, 2); % xor does module 2 operation
   reg = [reg(2:bw) xor_out];
   z(i+1) = sum(reg.*2.^(numel(reg)-1:-1:0)); % shift operation
   fprintf(out_file,'%d\n',z(i));
end

% Finish
fclose(out_file);
exit;

```



Verilog

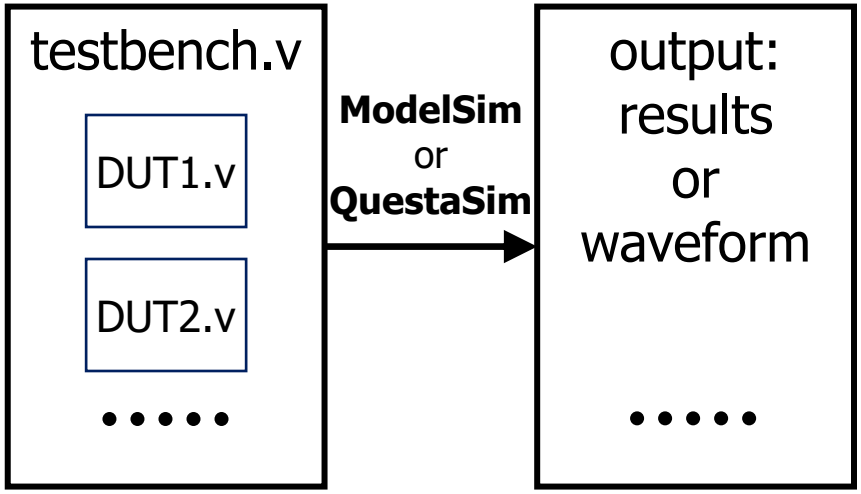
Hardware Description Language (HDL)

- An efficient way to design, verify, and test circuits
  - Custom design → Fine-grained but with low productivity
  - HDL → Higher development efficiency
- Popular HDLs: Verilog HDL and VHDL
- If you are not familiar with Verilog HDL, you can use the following tutorials:
  - http://www.verilogtutorial.info
  - http://www.doulos.com/knowhow/verilog_designers_guide
- Get started with the given example

Verilog HDL Simulator: QuestaSim

- Verilog simulator: QuestaSim/Modelsim by Mentor-Graphics
- Simulator files

1. Testbench file (testbench.v)
   - Defines input waveform
   - Defines output detection
2. Source files
   - RTL-level Verilog file (*.v)
   - Gate-level netlist (output of synthesis or APR)



- How to define timing constraints?
  - RTL-level: use "#xx" expression (does not affect the actual circuits)
  - Gate-level: include *.sdf file (output of synthesis or APR)

Verilog: lfsr1.v

```
1  `timescale 1ns/1ps
2
3  module lfsr1 (clk, resetn, seed, lfsr_out);
4
5      input clk, resetn;
6      input [15:0] seed;
7      output [15:0] lfsr_out;
8
9      reg [15:0] lfsr_out;
10
11     wire [15:0] lfsr_next;
12
```

- `timescale specifies the time unit and precision for the modules
  - Format: `timescale <time_unit>/<time_precision>
- input/output, wire/reg element definition
- Check the difference between “reg” and “wire” elements!

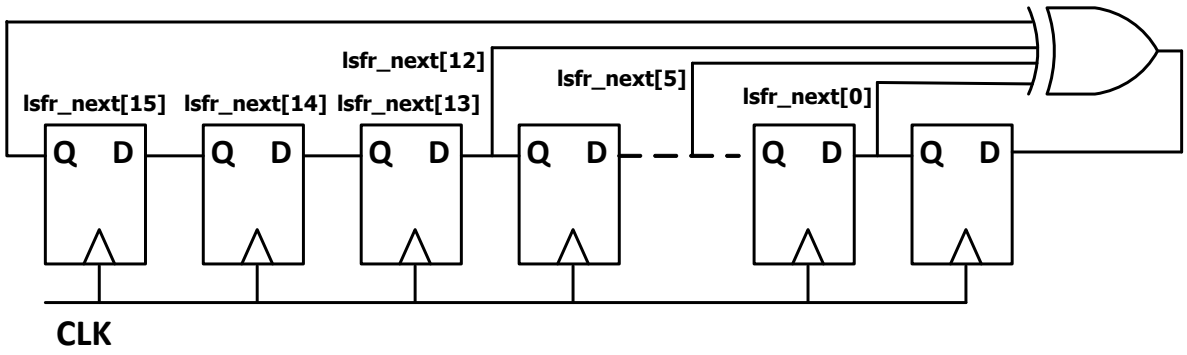
Verilog: lfsr1.v

```
13      always @(posedge clk) begin
14          if (~resetn) begin
15              lfsr_out <= #0.1 seed;
16          end
17          else begin
18              lfsr_out <= #0.1 lfsr_next;
19          end
20      end
21
22      assign lfsr_next = {lfsr_out[14:0], lfsr_out[15]^lfsr_out[12]^lfsr_out[5]^lfsr_out[0]};
23
24  endmodule
```

synchronous reset

continuous statement

- 'always@': Behavioral presentation – Procedure statement
- It is better to use the separated element for non-blocking assignment (<=) (Up to you!)
- #0.1 is the artificial delay (to be ignored in logic synthesis)
  - Because time_unit is 1ns, the delay is 100ps



Verilog: tb_lfsr1.v

```
1  `timescale 1ns/1ps
2  `define HALF_CLOCK_PERIOD #2.0
3  `define QSIM_OUT_FN "./lfsr1_rtl.result"
4  `define MATLAB_OUT_FN "../../matlab/lfsr1/lfsr1_matlab.result"
5
6  module testbench();
7
8      reg clk;
9      reg resetn;
10     reg [15:0] seed;
11
12     integer lfsr_out_matlab;
13     integer lfsr_out_qsim;
14
15     wire [15:0] lfsr_out;
16
17     integer i;
18     integer ret_write;
19     integer ret_read;
20     integer qsim_out_file;
21     integer matlab_out_file;
22
23     integer error_count = 0;
24
25     lfsr1 lfsr_0 ( .clk(clk), .resetn(resetn), .seed(seed), .lfsr_out(lfsr_out) );
26
27     always begin
28         `HALF_CLOCK_PERIOD;
29         clk = ~clk;
30     end
```

Device-Under-Test (DUT) instantiation

- **clk**: Using a 'reg' because it is used at the left-hand side of an always assignment
- **resetn, seed**: using a 'reg' in an initial block
- Clock period is 4 ns

Verilog: tb_lfsr1.v

```
32      initial begin
33          // File IO
34          qsim_out_file = $fopen(`QSIM_OUT_FN,"w");
35          if (!qsim_out_file) begin
36              $display("Couldn't create the output file.");
37              $finish;
38          end
39
40          matlab_out_file = $fopen(`MATLAB_OUT_FN,"r");
41          if (!matlab_out_file) begin
42              $display("Couldn't open the Matlab file.");
43              $finish;
44          end
45
46          // register setup
47          clk = 0;
48          resetn = 0;
49          seed = 16'd1;
50          @(posedge clk);
51
52          @(negedge clk);   // release resetn
53          resetn = 1;
```

- **$fopen** opens a disk file
  - QSIM_OUT_FN file is opened for writing
  - MATLAB_OUT_FN file is opened for reading
- Initial 'clk'=0, 'resetn'=0, 'seed'=1
  - **clk** signal toggles every HALF_CLOCK_PERIOD
  - At the first falling edge of **clk**, **resetn** becomes 1

Verilog: tb_lfsr1.v

```
55          @(posedge clk);   // start the first cycle
56          for (i=0 ; i<256; i=i+1) begin
57              // compare w/ the results from Matlab sim
58              ret_read = $fscanf(matlab_out_file, "%d", lfsr_out_matlab);
59              lfsr_out_qsim = lfsr_out;
60
61              $fwrite(qsim_out_file, "%0d\n", lfsr_out_qsim);
62              if (lfsr_out_qsim != lfsr_out_matlab) begin
63                  error_count = error_count + 1;
64              end
65
66              @(posedge clk);  // next cycle
67          end
68
69          // Any mismatch b/w rtl and matlab sims?
70          if (error_count > 0) begin
71              $display("The results DO NOT match with those from Matlab :( ");
72          end
73          else begin
74              $display("The results DO match with those from Matlab :) ");
75          end
76
77          // finishing this testbench
78          $fclose(qsim_out_file);
79          $fclose(matlab_out_file);
80          $finish;
81      end
82
83  endmodule // testbench
```

- From the second rising edge of **clk**, for loop statement starts
  - You can use files by using **$fscanf**, **$fwrite**
  - At the next clock's rising edge, it starts new comparison
- In this testbench, 256 comparisons are repeated

How to Run Simulation

- Command: $vsim –do “runsim.do”
  - **vsim** command is used to invoke the VSIM simulator
  - ‘-do’: instructs vsim to use the command specified by “cmd” or the “file”
  - Format: vsim [-do “cmd” | <file>]

- You can use shell scripts or Makefile

How to Run Simulation: runsim.do

```
1  ################################################
2  #  Modelsim do file to run simuilation
3  #  MS 7/2015
4  ################################################
5
6  vlib work
7  vmap work work
8
9  # include netlist and testbench files
10 vlog -incr ../../rtl/lfsr1/lfsr1.v
11 vlog -incr tb_lfsr1.v
12
13 # run simulation
14 vsim -t ps -lib work testbench
15 do waveformat.do
16 run -all
```

- **vlib** command creates a design library
- **vmap** command defines a mapping between a logical library name and a directory
  - Format: vmap [<logical_name>] [<path>]
- **vlog** command compiles Verilog source code into a specified working library
  - '-incr': performs an incremental compile

Reference: https://users.ece.cmu.edu/~kbiswas/modelsim/se_man.pdf

How to Run Simulation: runsim.do

```
1  ################################################
2  #  Modelsim do file to run simuilation
3  #  MS 7/2015
4  ################################################
5
6  vlib work
7  vmap work work
8
9  # include netlist and testbench files
10 vlog -incr ../../rtl/lfsr1/lfsr1.v
11 vlog -incr tb_lfsr1.v
12
13 # run simulation
14 vsim -t ps -lib work testbench
15 do waveformat.do
16 run -all
```

- **vsim** command
  - '-t': Specifies the simulator time resolution
  - '-lib': Specifies the default working library
  - Format: vsim [-arguments] [<library_name>.<design_unit>]
- **do** command executes commands contained in a macro file
- **run** command advances the simulation

Reference: https://users.ece.cmu.edu/~kbiswas/modelsim/se_man.pdf

How to Run Simulation: waveformat.do

```
onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate /testbench/clk
add wave -noupdate /testbench/resetn
add wave -noupdate /testbench/i
add wave -noupdate -radix unsigned /testbench/lfsr_out
add wave -noupdate -radix unsigned /testbench/lfsr_out_matlab
add wave -noupdate /testbench/lfsr_out_qsim
add wave -noupdate -radix unsigned /testbench/error_count
```

- **onerror** commands specifies one or more commands to be executed when a running macro encounters an error
- **add wave** commands adds the following objects to the window
- **WaveActiveNextPane** commands create a pane which contains signals

Reference: https://users.ece.cmu.edu/~kbiswas/modelsim/se_man.pdf

How to Run Simulation: waveformat.do

```
10 TreeUpdate [SetDefaultTree]
11 WaveRestoreCursors {{Cursor 1} {3 ns} 0}
12 quietly wave cursor active 1
13 configure wave -namecolwidth 223
14 configure wave -valuecolwidth 89
15 configure wave -justifyvalue left
16 configure wave -signalnamewidth 0
17 configure wave -snapdistance 10
18 configure wave -datasetprefix 0
19 configure wave -rowmargin 4
20 configure wave -childrowmargin 2
21 configure wave -gridoffset 0
22 configure wave -gridperiod 1
23 configure wave -griddelta 40
24 configure wave -timeline 0
25 configure wave -timelineunits ps
26 update
27 WaveRestoreZoom {0 ns} {12 ns}
```

- **TreeUpdate** command refreshes waveforms
- **WaveRestoreCursors** command restores any cursors you set during the original simulation
- **WaveRestoreZoom** command restores the Zoom range you set
- **Configure wave** commands are used only in saved Wave format files

Reference: https://users.ece.cmu.edu/~kbiswas/modelsim/se_man.pdf

Synthesis:
Design Compiler (DC)

Overview of Design Compiler (DC)

- Translate *.v file (RTL code) to *.nl.v file (gate-level netlist)
  - You can name the gate-level netlist your own way, if you maintain the extension as .v
- Inputs of Design Compiler
  - RTL code of your design (*.v): lfsr1.v
  - Standard cell library (*.db, *.sldb): common.tcl
  - Timing constraints (*.tcl): timing.tcl
  - Synthesis flow commands/setup (*.tcl): lfsr1.tcl
- Outputs of Design Compiler
  - Gate level netlist (*.nl.v)
  - Timing information (*.sdf)
  - Synthesis report (*.rpt)

Four Steps to Run Synthesis

1. Read design and library
2. Set design constraints
   - Timing constraints
   - Loading constraints
   - Other constraints (e.g. set_max_fanout)
3. Compile
4. Analyze and generate reports

- These 4 steps are organized in *.tcl file

Synthesis: lfsr1.tcl

```
 6 ################################################
 7 # Read design and library
 8 ################################################
 9
10 # Set the top_level name
11 set top_level lfsr1
12
13 # In this file, libray path and libraries which are used are defined
14 source -verbose "../common_scripts/common.tcl"
15
16 # Read verilog files
17 read_verilog {../../rtl/$top_level/$top_level.v}
18
19 # List the names of the designs loaded in memory
20 # Note: This command don't have any functionality for synthesis
21 #       This is for debugging
22 list_designs
23
24 # Check errors
25 if { [check_error -v] == 1 } { exit 1 }
26
27 # Set the current design
28 # Note: The active design is called the current design
29 #       Most commands are specific to the current design
30 current_design $top_level
31
32 # Link the design
33 # Note: The design must be connected to all the library components and designs if references
34 #       For each subdesign, a reference and a link must exist between the subdesgin and a design or component
35 #       in the link libraries
36 link
```

- Step 1: Read design and library

Library file: common.tcl

```
 3 # Set library paths
 4 # Note: DC uses the search path defined in the search_path variable to locate the library files
 5 set search_path [list "." "/courses/ee6321/share/ibm13rflpvt/synopsys/"]
 6
 7 # Set the DesignWare library
 8 # Note: A DesignWare library is a collection of reusable circuit-design building blocks that are tightly integrated
 9 #       into the Synopsys synthesis environment
10 set synthetic_library [list "dw_foundation.sldb"]
11
12 # Set the link library
13 # Note: DC uses the link library to resolve references
14 #       When you load a design into memory, DC also loads all libraries specified in the link_library variable
15 #       (For macros like RAM, ROM, PAD etc..)
16 set link_library [list "*" \
17                        "scx3_cmos8rf_lpvt_tt_1p2v_25c.db" \
18                        "dw_foundation.sldb"]
19
20 # Set the target library
21 # Note: DC uses the target library to build a circuit
22 #       During mapping, it selects functionally correct gates from the target library
23 set target_library "/courses/ee6321/share/ibm13rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_1p2v_25c.db"
```

Related to basic HDL operators

- If you have your own standard cells *.db file (Lab03), you can add it here to tell the design compiler to use them for mapping
- You can also use **set_dont_use CELLNAME** to tell the design compiler not to use certain cells when mapping

Link Library vs. Target Library

- Link library
  - Used to resolve the cell reference
  - When you refer to some cells from another part of the design or from the technology library, you need to link the library to use the cells
- Target library
  - Used to map the behavioral model to the structural model
  - Design Compiler uses this library to select cells for optimization and re-mapping
- Example

```
module example (clk, in1, in2, in3, out1, out2);
    ......
    ibm_sram sram_0 (.clk(clk), .in1(in1), .in2(in2), .out1(out1));
    assign out2 = in2 | in3;
    ......
endmodule
```

If you use the cell provided from Foundry, you need to define the 'link_library' to refer the cell

The 'target_library' should be needed to map this behavioral expression to the structural model

Synthesis: lfsr1.tcl

```
39 ################################################
40 # Define design rule constraints
41 ################################################
42
43 # Set maximum fanout of gates
44 # Note: This command sets the maximum allowable fanout load for the listed input ports
45 #       The object lists specifies a list of input ports and/or designs on which the max_fanout attribute is to be set
46 set_max_fanout 4 $top_level
47 set_max_fanout 4 [all_inputs]
48
49 # Set a maximum capacitance for the nets attached to named ports or to all the nets in a design
50 set_max_capacitance 0.005 [all_inputs]
51
52 # Verify the design consistency
53 # Note: This command reports an error that DC cannot resolve or a warning
54 check_design
55
56
57 ################################################
58 # Define design optimization constraints
59 ################################################
60
61 # Set timing environment through another .tcl file
62 source -verbose "./timing.tcl"
63
64 # Set the fix_multiple_port_nets attribute to a specified value on the current design or a list of designs
65 # Note: '-all' option inserts buffers to the ports
66 #       '-buffer_constants' option inserts buffers to logic constants instead of duplicating them
67 set_fix_multiple_port_nets -all -buffer_constants
```

- Step 2: Set design constraints
  - Timing constraints
  - Loading constraints
  - Other constraints (e.g. set_max_fanout)

Constraint file: timing.tcl

```
# Time unit: 1ns (in the IBM power-delivery library)
# Capacitance unit: 1pF (in the IBM power-delivery library)

################################################
# Define the clock timing
################################################

# Variables for clock
set clk_period 4
set clk_uncertainty 0.01
set clk_transition 0.01

set clk_name "clk"
set clk_port "clk"

# Create a clock if clock port is found
if {[sizeof_collection [get_ports $clk_port]] > 0} {

    # Create a clock object and define its waveform in the current design
    # Note: source_objects are the pin or port where the clock waveform is applied to the design
    #       If no waveform is specified, 50% duty cycle is assumed
    create_clock $clk_port -name $clk_name -period $clk_period

    # Set the rise_drive or fall_drive attributes to the specified resistance values on the specified input and inout ports
    # Note: For heavily loaded driving ports, such as clock lines, keep the drive strength seeeting 0 so that DC does not buffer the net
    set_drive 0 [get_clocks $clk_name]
}

# Create virtual clock if clock port is not found
if {[sizeof_collection [get_ports $clk_port]] == 0} {

    # Create a clock object and define its waveform in the current design
    # Note: source_objects are the pin or port where the clock waveform is applied to the design
    #       If no waveform is specified, 50% duty cycle is assumed
    set clk_name vclk
    create_clock -period $clk_period -name vclk

    # Set the rise_drive or fall_drive attributes to the specified resistance values on the specified input and inour ports
    # Note: For heavily loaded driving ports, such as clock lines, keep the drive strength seeeting 0 so that DC does not buffer the net
    set_drive 0 [get_clocks $clk_name]
}
```

- Virtual clock can be used for synthesizing combinational logics

Constraint file: timing.tcl

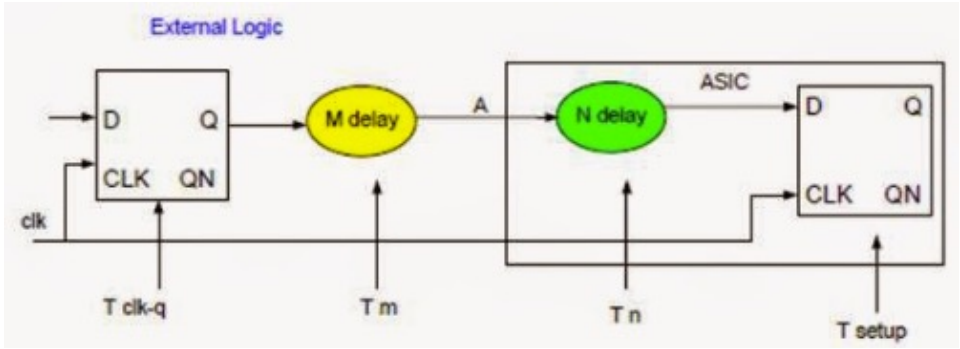
```
45 # Set clock uncertainty (skew)
46 # Note: This command considers different clock arrival time to FFs
47 set_clock_uncertainty $clk_uncertainty [get_clocks $clk_name]
48
49 # Set the transition time at the clock pins of all sequential devices clocked by the specified ideal clocks
50 set_clock_transition $clk_transition [get_clocks $clk_name]
51
52 # Fix hold violations at registers during compilation
53 # Note: If fix_hold or min_delay is specified, the minimum delay is a secondary optimization cost
54 #       DC inserts more buffers to solve the hold violations
55 set_fix_hold [all_clocks]
56
57 # Prevent cells and nets from being modified or replaced during optimization
58 # Note: Placing a dont_touch_network on a clock object prevents the toll from buffering the clock network
59 #       If dont_touch_network is set, DC reports violations (e.g. fanout) but does not modify the network during optimization
60 set_dont_touch_network $clk_port
61
62 # Mark ports and pins as sources of ideal networks
63 # Note: This command disables the timing update and optimization of cells and nets in the fanout of the specified objects
64 set_ideal_network      $clk_port
```

- Define the clock timing

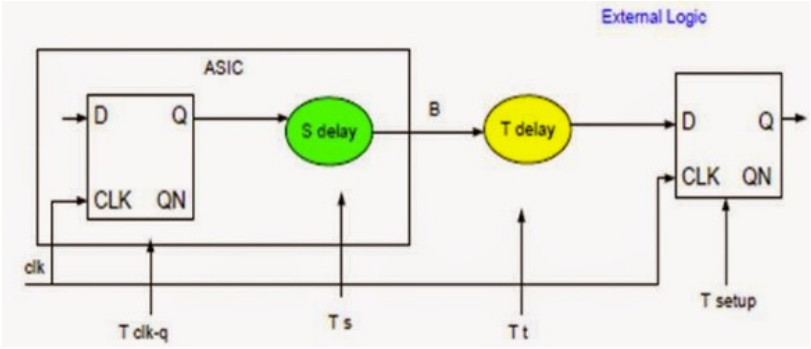
Constraint file: timing.tcl

```
67 ################################################
68 # Define the input and output ports' timing
69 ################################################
70
71 # Varaibles for input/output delay
72 set typical_input_delay 0.1
73 set typical_output_delay 0.1
74 set typical_wire_load 0.005
75
76 # Specifiy drive characteristics on ports that are driven by cells in the technology library
77 set_driving_cell -lib_cell INVX1TS [all_inputs]
78
79 # Define the arrival time for input ports
80 # Note: We define the input delay constraint relative to the system clock and to the other inputs
81 #       If data is stable (A) after the clock edge, input_delay is (A) (related to setup time)
82 set_input_delay $typical_input_delay [all_inputs] -clock $clk_name
83
84 # Remove input delay on the specified pins or input ports
85 remove_input_delay -clock $clk_name [find port $clk_port]
86
87 # Set output delay on pins or output ports relative to a clock signal
88 # Note: If data needs to be available (B) before the clock edge, output_delay is (B)
89 #       MIN & MAX delays are related to the shorest and longest path, respectively
90 set_output_delay $typical_output_delay [all_outputs] -clock $clk_name
91
92 # Set a capacitive load value on input and output ports of the design
93 set_load 0.005 [all_outputs]
```

If **input_delay** is T_m, the logic (green circle) should be done within T_{clk} - T_m



If **output_delay** is T_t, the logic (green circle) should be done within T_{clk} - T_t



Synthesis: lfsr1.tcl

```
70 ################################################
71 # Optimize the design
72 ################################################
73
74 # Verify the design consistency
75 check_design
76
77 # Set the current design
78 current_design $top_level
79
80 # Link the design
81 link
82
83 # Synthesize the design
84 # Note : This command performs a high-effort compile on the current design for better quality of results (QoR)
85 compile_ultra
```

- Step 3: Compile

Synthesis: lfsr1.tcl

Step 4: Analyze and generate reports

```
 88 ################################################
 89 # Analyze and resolve design problems
 90 ################################################
 91
 92 # Verify the design consistency
 93 check_design
 94
 95 # Rename modules, signals according to the naming rules
 96 source -verbose "../common_scripts/namingrules.tcl"
 97
 98 # Generate a report file
 99 set rpt_file "${top_level}.dc.rpt"
100 set maxpaths 20
101
102 check_design >> ${rpt_file}
103
104 report_area  >> ${rpt_file}
105 report_power -hier -analysis_effort medium >> ${rpt_file}
106 report_design >> ${rpt_file}
107 report_cell >> ${rpt_file}
108 report_port -verbose >> ${rpt_file}
109 report_compile_options >> ${rpt_file}
110 report_constraint -all_violators -verbose >> ${rpt_file}
111 report_timing -path full -delay max -max_paths $maxpaths -nworst 100 >> ${rpt_file}
112 report_timing_requirements >> ${rpt_file}
```

```
121 ################################################
122 # Save the design database
123 ################################################
124
125 # Generate structural verilog netlist
126 write -hierarchy -format verilog -output "${top_level}.nl.v"
127
128 # Write a Standard Delay Format (SDF) file
129 # Note : SDF is an IEEE standard for the representation and interpretation of timing data
130 #        The SDF file includes delays (module path, device, interconnect, and port), timing checks (setup, hold, recovery,
131 #        skew, width, and period), timing constraints (path and skew), incremental and absolute delays, and so on
132 write_sdf "${top_level}.syn.sdf"
133
134 # Write a SDC file
135 # Note : SDC is a format used to specify the design intent, including the timing, power and area constraints for a design
136 write_sdc "${top_level}.syn.sdc" -version 1.7
137
138 # Finish synthesis
139 quit
```

How to Synthesize

- Command: $dc_shell –f "<filename>" | tee <output logfile>
  - When you use "tee" command, log messages are saved in the logfile
- You can use shell scripts or Makefile
- Output files
  - Gate level netlist (*.nl.v)
  - Timing information (*.sdf)
  - Synthesis report (*.rpt)

Synthesis Output File: lfsr1.nl.v

```
1  ////////////////////////////////////////////////////////////
2  // Created by: Synopsys DC Ultra(TM) in wire load mode
3  // Version   : O-2018.06-SP5-1
4  // Date      : Thu Feb 13 21:40:19 2020
5  ////////////////////////////////////////////////////////////
6
7
8  module lfsr1 ( clk, resetn, seed, lfsr_out );
9    input [15:0] seed;
10   output [15:0] lfsr_out;
11   input clk, resetn;
12   wire   N3, N4, N5, N6, N7, N8, N9, N10, N11, N12, N13, N14, N15, N16, N17,
13          N18, n50, n60, n70, n80, n90, n100, n110, n120, n130, n140, n150,
14          n160, n170, n180, n19;
15
16   DFFQX1TS lfsr_out_reg_1_ ( .D(N4), .CK(clk), .Q(lfsr_out[1]) );
17   DFFQX1TS lfsr_out_reg_2_ ( .D(N5), .CK(clk), .Q(lfsr_out[2]) );
18   DFFQX1TS lfsr_out_reg_3_ ( .D(N6), .CK(clk), .Q(lfsr_out[3]) );
19   DFFQX1TS lfsr_out_reg_4_ ( .D(N7), .CK(clk), .Q(lfsr_out[4]) );
20   DFFQX1TS lfsr_out_reg_6_ ( .D(N9), .CK(clk), .Q(lfsr_out[6]) );
21   DFFQX1TS lfsr_out_reg_7_ ( .D(N10), .CK(clk), .Q(lfsr_out[7]) );
22   DFFQX1TS lfsr_out_reg_8_ ( .D(N11), .CK(clk), .Q(lfsr_out[8]) );
23   DFFQX1TS lfsr_out_reg_9_ ( .D(N12), .CK(clk), .Q(lfsr_out[9]) );
24   DFFQX1TS lfsr_out_reg_10_ ( .D(N13), .CK(clk), .Q(lfsr_out[10]) );
25   DFFQX1TS lfsr_out_reg_11_ ( .D(N14), .CK(clk), .Q(lfsr_out[11]) );
26   DFFQX1TS lfsr_out_reg_13_ ( .D(N16), .CK(clk), .Q(lfsr_out[13]) );
27   DFFQX1TS lfsr_out_reg_14_ ( .D(N17), .CK(clk), .Q(lfsr_out[14]) );
28   DFFQX1TS lfsr_out_reg_15_ ( .D(N18), .CK(clk), .Q(lfsr_out[15]) );
29   DFFQX1TS lfsr_out_reg_0_ ( .D(N3), .CK(clk), .Q(lfsr_out[0]) );
30   DFFQX1TS lfsr_out_reg_5_ ( .D(N8), .CK(clk), .Q(lfsr_out[5]) );
31   DFFQX1TS lfsr_out_reg_12_ ( .D(N15), .CK(clk), .Q(lfsr_out[12]) );
```

- This module is composed of standard cells provided by the Foundry

Synthesis Output File: lfsr1.syn.sdf

```
(DELAYFILE
(SDFVERSION "OVI 2.1")
(DESIGN "lfsr1")
(DATE "Thu Feb 13 21:40:19 2020")
(VENDOR "scx3_cmos8rf_lpvt_tt_1p2v_25c")
(PROGRAM "Synopsys Design Compiler cmos")
(VERSION "O-2018.06-SP5-1")
(DIVIDER /)
(VOLTAGE 1.20:1.20:1.20)
(PROCESS "tt_1p2v_25c")
(TEMPERATURE 25.00:25.00:25.00)
(TIMESCALE 1ns)
(CELL
  (CELLTYPE "lfsr1")
  (INSTANCE)
  (DELAY
    (ABSOLUTE
    (INTERCONNECT U49/Y U53/A (0.000:0.000:0.000))
    (INTERCONNECT U49/Y U52/A (0.000:0.000:0.000))
    (INTERCONNECT U49/Y U51/A (0.000:0.000:0.000))
    (INTERCONNECT U49/Y U50/A (0.000:0.000:0.000))
    (INTERCONNECT U45/Y U49/A (0.000:0.000:0.000))
    (INTERCONNECT U44/Y U48/A (0.000:0.000:0.000))
    (INTERCONNECT U44/Y U47/A (0.000:0.000:0.000))
    (INTERCONNECT U44/Y U46/A (0.000:0.000:0.000))
    (INTERCONNECT U44/Y U45/A (0.000:0.000:0.000))
    (INTERCONNECT resetn U44/A (0.000:0.000:0.000))
    (INTERCONNECT U42/Y U43/A (0.000:0.000:0.000))
    (INTERCONNECT U46/Y U42/A (0.000:0.000:0.000))
    (INTERCONNECT U45/Y U41/A0 (0.000:0.000:0.000))
    (INTERCONNECT lfsr_out_reg_0_/Q U41/A1 (0.000:0.000:0.000))
    (INTERCONNECT U51/Y U41/B0 (0.000:0.000:0.000))
    (INTERCONNECT seed[1] U41/B1 (0.000:0.000:0.000))
    (INTERCONNECT U45/Y U40/A0 (0.000:0.000:0.000))
    (INTERCONNECT lfsr_out_reg_1_/Q U40/A1 (0.000:0.000:0.000))
    (INTERCONNECT U52/Y U40/B0 (0.000:0.000:0.000))
    (INTERCONNECT seed[2] U40/B1 (0.000:0.000:0.000))
    (INTERCONNECT U46/Y U39/A0 (0.000:0.000:0.000))
```

min:typ:max setting

Interconnect delay

```
(CELL
  (CELLTYPE "INVX2TS")
  (INSTANCE U53)
  (DELAY
    (ABSOLUTE
    (IOPATH A Y (0.132:0.132:0.132) (0.099:0.099:0.099))
    )
  )
)
(CELL
  (CELLTYPE "INVX2TS")
  (INSTANCE U52)
  (DELAY
    (ABSOLUTE
    (IOPATH A Y (0.132:0.132:0.132) (0.099:0.099:0.099))
    )
  )
)
(CELL
  (CELLTYPE "INVX2TS")
  (INSTANCE U51)
  (DELAY
    (ABSOLUTE
    (IOPATH A Y (0.132:0.132:0.132) (0.099:0.099:0.099))
    )
  )
)
(CELL
  (CELLTYPE "INVX2TS")
  (INSTANCE U50)
  (DELAY
    (ABSOLUTE
    (IOPATH A Y (0.132:0.132:0.132) (0.099:0.099:0.099))
    )
  )
)
```

Cell delay

r_delay f_delay

- It includes delay information
- It will be used in post-synthesis **dynamic** timing verification

Synthesis Output File: lfsr1.dc.rpt

```
 3 ****************************************
 4 Report : area
 5 Design : lfsr1
 6 Version: O-2018.06-SP5-1
 7 Date   : Thu Feb 13 21:40:19 2020
 8 ****************************************
 9
10 Information: Updating design information... (UID-85)
11 Library(s) Used:
12
13     scx3_cmos8rf_lpvt_tt_1p2v_25c (File: /courses/ee6321/share/ibm13rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_1p2v_25c.db)
14
15 Number of ports:                           34
16 Number of nets:                            65
17 Number of cells:                           47
18 Number of combinational cells:             31
19 Number of sequential cells:                16
20 Number of macros/black boxes:               0
21 Number of buf/inv:                         12
22 Number of references:                       5
23
24 Combinational area:                249.120002
25 Buf/Inv area:                       53.280002
26 Noncombinational area:             391.679993
27 Macro/Black Box area:                0.000000
28 Net Interconnect area:      undefined  (No wire load specified)
29
30 Total cell area:                   640.799995
31 Total area:                 undefined
32 1
```

- Area analysis
  - This is just area estimation from synthesis
  - It doesn’t consider actual interconnect and floorplan

Synthesis Output File: lfsr1.dc.rpt

```
****************************************
Report : power
        -hier
        -analysis_effort medium
Design : lfsr1
Version: O-2018.06-SP5-1
Date   : Thu Feb 13 21:40:19 2020
****************************************


Library(s) Used:

    scx3_cmos8rf_lpvt_tt_1p2v_25c (File: /courses/ee6321/share/ibm13rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_1p2v_25c.db)


Operating Conditions: tt_1p2v_25c   Library: scx3_cmos8rf_lpvt_tt_1p2v_25c
Wire Load Model Mode: top


Global Operating Voltage = 1.2
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


--------------------------------------------------------------------------------
                                   Switch   Int      Leak     Total
Hierarchy                          Power    Power    Power    Power    %
--------------------------------------------------------------------------------
lfsr1                              8.93e-03    0.113  808.625    0.122 100.0
1
```

- Power analysis
  - No information about the inputs -> inaccurate

Synthesis Output File: lfsr1.dc.rpt

```
501 ****************************************
502 Report : timing
503         -path full
504         -delay max
505         -nworst 100
506         -max_paths 20
507 Design : lfsr1
508 Version: O-2018.06-SP5-1
509 Date   : Thu Feb 13 21:40:19 2020
510 ****************************************
511
512 Operating Conditions: tt_1p2v_25c   Library: scx3_cmos8rf_lpvt_tt_1p2v_25c
513 Wire Load Model Mode: top
514
515   Startpoint: lfsr_out_reg_5_
516               (rising edge-triggered flip-flop clocked by clk)
517   Endpoint: lfsr_out_reg_0_
518             (rising edge-triggered flip-flop clocked by clk)
519   Path Group: clk
520   Path Type: max
521
522   Point                                    Incr       Path
523   -----------------------------------------------------------
524   clock clk (rise edge)                    0.00       0.00
525   clock network delay (ideal)              0.00       0.00
526   lfsr_out_reg_5_/CK (DFFQX1TS)            0.00       0.00 r
527   lfsr_out_reg_5_/Q (DFFQX1TS)             0.74       0.74 f
528   U28/Y (XOR2XLTS)                         0.33       1.07 r
529   U26/Y (XOR2XLTS)                         0.41       1.48 f
530   U25/Y (AO22XLTS)                         0.44       1.92 f
531   lfsr_out_reg_0_/D (DFFQX1TS)             0.00       1.92 f
532   data arrival time                                   1.92
533
534   clock clk (rise edge)                    4.00       4.00
535   clock network delay (ideal)              0.00       4.00
536   clock uncertainty                       -0.01       3.99
537   lfsr_out_reg_0_/CK (DFFQX1TS)            0.00       3.99 r
538   library setup time                      -0.33       3.66
539   data required time                                  3.66
540   -----------------------------------------------------------
541   data required time                                  3.66
542   data arrival time                                  -1.92
543
544   slack (MET)                                         1.74
```

Timing analysis

It shows individual contribution to path delay

clock period = 4 ns

Slack: Negative indicates a violation
Make sure it MET!

After Synthesis…

- Post-synthesis dynamic timing verification
  - CAD tool: ModelSim
  - Compare with RTL-level verification, what's the difference?

- Post-synthesis static timing and power verification
  - CAD tool: PrimeTime
  - What you get:
    - the worst-case delay path
    - power consumption with detailed switching activities
    - setup/hold time violations
  - DC also provides these information, but inaccurate

Verilog Simulation w/ Back Annotation

- qsim_rtl/tb_lfsr1/runsim.do

```
1  ##############################################
2  #  Modelsim do file to run simulation
3  #  MS 7/2015
4  ##############################################
5
6  vlib work
7  vmap work work
8
9  # include netlist and testbench files
10 vlog -incr ../../rtl/lfsr1/lfsr1.v
11 vlog -incr tb_lfsr1.v
12
13 # run simulation
14 vsim -t ps -lib work testbench
15 do waveformat.do
16 run -all
```

RTL code

- qsim_dc/tb_lfsr1/runsim.do

```
1  ##############################################
2  #  Modelsim do file to run simulation
3  #  MS 7/2015
4  ##############################################
5
6  vlib work
7  vmap work work
8
9  # include netlist and testbench files
10 vlog -incr /courses/ee6321/share/ibm13rflpvt/verilog/ibm13rflpvt.v
11 vlog -incr ../../dc/lfsr1/lfsr1.nl.v
12 vlog -incr tb_lfsr1.v
13
14 # run simulation
15 vsim -t ps -lib work testbench
16 do waveformat.do
17 run -all
```

Gate-level netlist &
std. cell library

- For simple verification, you can share the same testbench

Verilog Simulation wi/ Back Annotation

- qsim_dc/tb_lfsr1/tb_lfsr1.v

```
25      lfsr1 lfsr_0 ( .clk(clk), .resetn(resetn), .seed(seed), .lfsr_out(lfsr_out) );
26
27      initial $sdf_annotate("../../dc/lfsr1/lfsr1.syn.sdf", lfsr_0);
28
29      always begin
30          `HALF_CLOCK_PERIOD;
31          clk = ~clk;
32      end
33
34      initial begin
35          // File IO
36          qsim_out_file = $fopen(`QSIM_OUT_FN,"w");
37          if (!qsim_out_file) begin
38              $display("Couldn't create the output file.");
39              $finish;
40          end
41
42          matlab_out_file = $fopen(`MATLAB_OUT_FN,"r");
43          if (!matlab_out_file) begin
44              $display("Couldn't open the Matlab file.");
45              $finish;
46          end
47
48          $dumpfile("./lfsr1.vcd");
49          $dumpvars(0, testbench.lfsr_0);
```

Add delay information

Create the .vcd file

```
82          // finishing this testbench
83          $fclose(qsim_out_file);
84          $fclose(matlab_out_file);
85
86          $dumpall;
87          $dumpflush;
88
89          $finish;
```

- Instead of using $sdf_annotate, you can change "runsim.do"
  - vsim +acc –t ps –lib work –sdftyp lfsr_0=../../dc/lfsr1/lfsr1.syn.sdf testbench

Verilog Simulation wi/ Back Annotation

- qsim_rtl/tb_lfsr1/tb_lfsr1.v

```
48        $dumpfile("./lfsr1.vcd");
49        $dumpvars(0, testbench.lfsr_0);
```

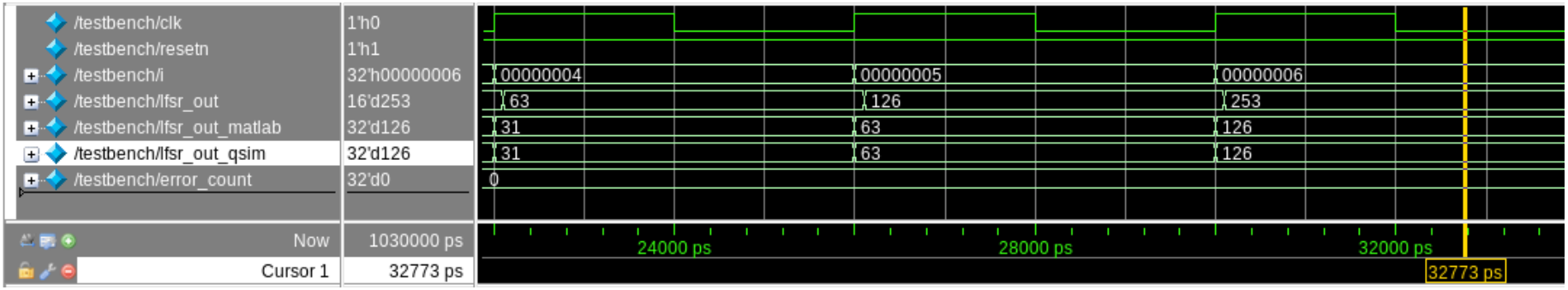
Create the .vcd file

```
82        // finishing this testbench
83        $fclose(qsim_out_file);
84        $fclose(matlab_out_file);
85
86        $dumpall;
87        $dumpflush;
88
89        $finish;
```

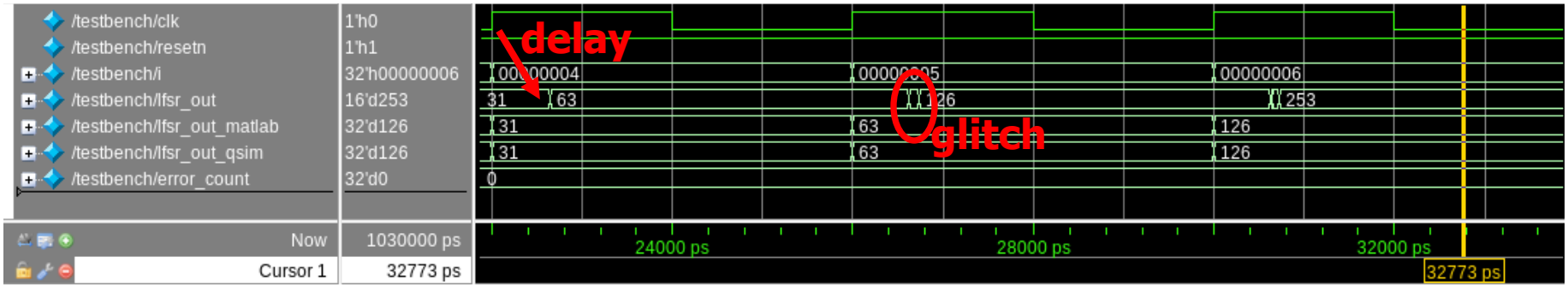
- Look up the usage of $dumpvars command
- .vcd file
  - contains the switching activity for the inputs provided
  - will be used to obtain an estimate of power using PrimeTime
- We can obtain realistic estimate of power given proper inputs

Verilog Simulation wi/ Back Annotation

- RTL code simulation



- Gate-level netlist simulation



- If you cannot observe delay, try to change <time_precision> of `timescale in the Verilog file

PrimeTime

Post-synthesis Verification wi/ PrimeTime

- Inputs of PrimeTime
  - Timing constraints (*.tcl): timing.tcl
  - PrimeTime commands/setup (*.tcl): lfsr1.tcl
- Outputs of PrimeTime
  - Timing information (*.sdf)
  - PrimeTime report (*.rpt)
- Go back to compile phase if any violations occurs
- Check power, timing
- .db file
  - The .db file is compiled version of .lib file
  - It contains
    - units that will be used by the tool for time, voltage, etc.
    - the operating conditions
    - models for timing and power of the standard cells

PrimeTime: lfsr1.tcl

```
51 # Generate a report file
52 set rpt_file "${top_level}.pt.rpt"
53 check_timing
54
55 report_design >> ${rpt_file}
56 report_reference >> ${rpt_file}
57 report_constraint >> ${rpt_file}
58 report_constraint -all_violators -significant_digits 4 >> ${rpt_file}
59 report_timing -significant_digits 4 -delay_type min_max >> ${rpt_file}
60
61 # Do power analysis
62 set power_analysis_mode "time_based"
63 read_vcd "../../qsim_dc/tb_lfsr1/lfsr1.vcd" -strip_path "testbench/lfsr_0"
64 report_switching_activity >> ${rpt_file}
65 report_switching_activity -list_not_annotated >> ${rpt_file}
66 update_power
67 report_power >> ${rpt_file}
68 report_power -hierarchy >> ${rpt_file}
69
70 write_sdf -context verilog "./lfsr1.sdf"
71
72 # Finish PrimeTime
73 quit
```

- The PrimeTime script is almost similar to the DC script except for analysis parts
- Clock period here is the same as the Verilog testbench where you generate the .vcd file!!

How to Run PrimeTime

- Command: $pt_shell –file "<filename>" | tee <output logfile>
  - When you use "tee" command, log messages are saved in the logfile
- You can use shell scripts or Makefile
- Output files
  - Timing information (*.sdf)
  - Synthesis report (*.rpt)

Man Commands

- If you want to know detail information about DC or PrimeTime commands…
- Run tools without .tcl scripts
  - $dc_shell (DC) / $pt_shell (PrimeTime)
  - Then, you can also type all commands after running tools instead of using .tcl scripts (not-effective)
- Type $man (command name)

```
dc_shell> man set_max_fanout
2.  Synopsys Commands                                   Command Reference
                              set_max_fanout

NAME
       set_max_fanout
              Sets  the max_fanout attribute to a specified value on specified
              input ports and designs.

SYNTAX
       status set_max_fanout
               fanout_value
               object_list

ARGUMENTS
       fanout_value
              Specifies the value to which the max_fanout attribute is  to  be
              set; that is, the maximum fanout value.

       object_list
              Specifies  a  list  of  input  ports and/or designs on which the
              max_fanout attribute is to be set.
```

PrimeTime Output File: lfsr1.pt.rpt

- Timing analysis (Static timing analysis)

```
142  Startpoint: seed[0] (input port clocked by clk)
143  Endpoint: lfsr_out_reg_0_
144            (rising edge-triggered flip-flop clocked by clk)
145  Path Group: clk
146  Path Type: min
147
148  Point                                   Incr       Path
149  -----------------------------------------------------------
150  clock clk (rise edge)                 0.0000     0.0000
151  clock network delay (ideal)           0.0000     0.0000
152  input external delay                  0.1000     0.1000 r
153  seed[0] (in)                          0.0134     0.1134 r
154  U25/Y (AO22XLTS)                      0.2236     0.3369 r
155  lfsr_out_reg_0_/D (DFFQX1TS)          0.0000     0.3369 r
156  data arrival time                                0.3369
157
158  clock clk (rise edge)                 0.0000     0.0000
159  clock network delay (ideal)           0.0000     0.0000
160  clock reconvergence pessimism         0.0000     0.0000
161  clock uncertainty                     0.0100     0.0100
162  lfsr_out_reg_0_/CK (DFFQX1TS)                    0.0100 r
163  library hold time                    -0.0995    -0.0895
164  data required time                              -0.0895
165  -----------------------------------------------------------
166  data required time                              -0.0895
167  data arrival time                               -0.3369
168  -----------------------------------------------------------
169  slack (MET)                                      0.4264
```

Minimum delay

```
173  Startpoint: lfsr_out_reg_5_
174              (rising edge-triggered flip-flop clocked by clk)
175  Endpoint: lfsr_out_reg_0_
176              (rising edge-triggered flip-flop clocked by clk)
177  Path Group: clk
178  Path Type: max
179
180  Point                                                 Incr       Path
181  ---------------------------------------------------------------------------
182  clock clk (rise edge)                               0.0000     0.0000
183  clock network delay (ideal)                         0.0000     0.0000
184  lfsr_out_reg_5_/CK (DFFQX1TS)                       0.0000     0.0000 r
185  lfsr_out_reg_5_/Q (DFFQX1TS)                        0.7366     0.7366 f
186  U28/Y (XOR2XLTS)                                    0.3303     1.0669 r
187  U26/Y (XOR2XLTS)                                    0.4126     1.4795 f
188  U25/Y (AO22XLTS)                                    0.4399     1.9194 f
189  lfsr_out_reg_0_/D (DFFQX1TS)                        0.0000     1.9194 f
190  data arrival time                                              1.9194
191
192  clock clk (rise edge)                               4.0000     4.0000
193  clock network delay (ideal)                         0.0000     4.0000
194  clock reconvergence pessimism                       0.0000     4.0000
195  clock uncertainty                                  -0.0100     3.9900
196  lfsr_out_reg_0_/CK (DFFQX1TS)                                  3.9900 r
197  library setup time                                 -0.3327     3.6573
198  data required time                                             3.6573
199  ---------------------------------------------------------------------------
200  data required time                                             3.6573
201  data arrival time                                             -1.9194
202  ---------------------------------------------------------------------------
203  slack (MET)                                                    1.7379
```

Maximum delay

PrimeTime Output File: lfsr1.pt.rpt

- Switching activity analysis

```
 Switching Activity Overview Statistics for "lfsr1"
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
                  From Activity     From          From           From           From            From            Default (%)     Propagated(%)   Implied(%)      Not             Total
Object Type       File (%)          SSA (%)       SSA Force (%)  SSA Force (%)  SCA (%)         Clock (%)                                                       Annotated(%)
                                                  Annotated      Implied
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nets             65(100.00%)       0(0.00%)      0(0.00%)       0(0.00%)       0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        65
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Nets Driven by
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Primary Input     18(100.00%)       0(0.00%)      0(0.00%)       0(0.00%)       0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        18
Tri-State         0(0%)             0(0%)         0(0%)          0(0%)          0(0%)           0(0%)           0(0%)           0(0%)           0(0%)           0(0%)           0
Black Box         0(0%)             0(0%)         0(0%)          0(0%)          0(0%)           0(0%)           0(0%)           0(0%)           0(0%)           0(0%)           0
Sequential        16(100.00%)       0(0.00%)      0(0.00%)       0(0.00%)       0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        16
Combinational     31(100.00%)       0(0.00%)      0(0.00%)       0(0.00%)       0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        0(0.00%)        31
Memory            0(0%)             0(0%)         0(0%)          0(0%)          0(0%)           0(0%)           0(0%)           0(0%)           0(0%)           0(0%)           0
Clock Gate        0(0%)             0(0%)         0(0%)          0(0%)          0(0%)           0(0%)           0(0%)           0(0%)           0(0%)           0(0%)           0
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

- Power analysis (Testbench is used)

```
****************************************
Report : Time Based Power
Design : lfsr1
Version: P-2019.03-SP2
Date   : Fri Feb 14 01:37:40 2020
****************************************



  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User defined power group

                        Internal  Switching  Leakage    Total
Power Group             Power     Power      Power      Power    (     %)  Attrs
--------------------------------------------------------------------------------
clock_network           8.585e-05    0.0000    0.0000 8.585e-05 (63.87%)  i
register                2.772e-05 9.355e-06 5.545e-10 3.708e-05 (27.58%)
combinational           9.496e-06 1.999e-06 2.446e-10 1.150e-05 ( 8.55%)
sequential                 0.0000    0.0000    0.0000    0.0000 ( 0.00%)
memory                     0.0000    0.0000    0.0000    0.0000 ( 0.00%)
io_pad                     0.0000    0.0000    0.0000    0.0000 ( 0.00%)
black_box                  0.0000    0.0000    0.0000    0.0000 ( 0.00%)

  Net Switching Power  = 1.135e-05   ( 8.45%)
  Cell Internal Power  = 1.231e-04   (91.55%)
  Cell Leakage Power   = 7.991e-10   ( 0.00%)
                         ---------
Total Power            = 1.344e-04  (100.00%)

X Transition Power     =    0.0000
Glitching Power        =    0.0000

Peak Power             = 3.559e-03
Peak Time              =    350.000
```

End of the Slides